```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Loading dataset
df = pd.read_csv(r"./laptop_data.csv")
```

```
df.head()
```

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	
0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel I Grap
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel Grap 6
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel Grap

```
df.tail()
```

	Unnamed: 0	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory
1298	1298	Lenovo	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD
1299	1299	Lenovo	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD
1300	1300	Lenovo	Notebook	14.0	1366x768	Intel Celeron Dual Core N3050	2GB	64GB Flash Storage

```
#Checking the shape of dataset
df.shape
```

```
(1303, 12)
```

```
#Checking for the columns
df.columns
```

```
Index(['Unnamed: 0', 'Company', 'TypeName', 'Inches', 'ScreenResolution',
       'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight', 'Price'],
      dtype='object')
```

```
#Checking for the Data-Types of attributes
df.dtypes
```

```
Unnamed: 0            int64
Company              object
TypeName             object
Inches              float64
ScreenResolution     object
Cpu                  object
Ram                  object
Memory               object
Gpu                  object
OpSys                object
Weight               object
Price               float64
dtype: object
```

```
#Overall description of Dataset
df.describe()
```

	Unnamed: 0	Inches	Price
count	1303.00000	1303.000000	1303.000000
mean	651.00000	15.017191	59870.042910
std	376.28801	1.426304	37243.201786
min	0.00000	10.100000	9270.720000
25%	325.50000	14.000000	31914.720000
50%	651.00000	15.600000	52054.560000
75%	976.50000	15.600000	79274.246400
max	1302.00000	18.400000	324954.720000

```
#Checking for the duplicate rows
df.duplicated().sum()
```

```
0
```

There is no duplicate row in our dataset

```
#Checking for null-values
df.isnull().sum()
```

```
Unnamed: 0          0
Company             0
TypeName            0
Inches              0
ScreenResolution    0
Cpu                 0
Ram                 0
Memory              0
Gpu                 0
OpSys               0
Weight              0
Price               0
dtype: int64
```

There is no NULL values in any of the attributes

```
#Dropping the 'Unnamed: 0' columns bcz it is of no use:
df.drop(columns='Unnamed: 0', inplace=True)
```

```
df.head(2)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS

```
#Now removing the suffix GB and kg from the columns 'Ram' and 'Weight', so as to make their dtypes as float from object:
df['Ram'] = df['Ram'].str.replace('GB', '')
df['Weight'] = df['Weight'].str.replace('kg', '')
df.head(2)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS

```
df.dtypes
```

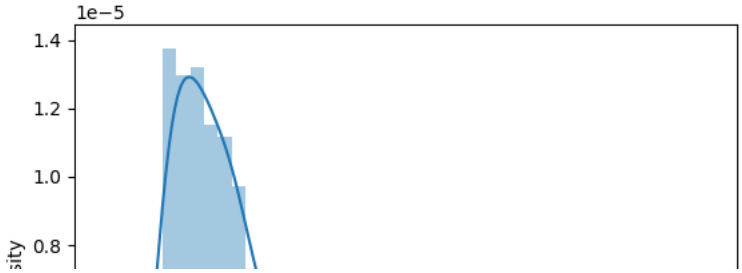
```
Company              object
TypeName             object
Inches              float64
ScreenResolution     object
Cpu                  object
Ram                  object
Memory               object
Gpu                  object
OpSys                object
Weight               object
Price               float64
dtype: object
```

```
df['Ram'] = df['Ram'].astype('int')
df['Weight'] = df['Weight'].astype('float')
df.dtypes
```

```
Company              object
TypeName             object
Inches              float64
ScreenResolution     object
Cpu                  object
Ram                   int64
Memory               object
Gpu                  object
OpSys                object
Weight              float64
Price               float64
dtype: object
```

```
sns.distplot(df['Price'])
```

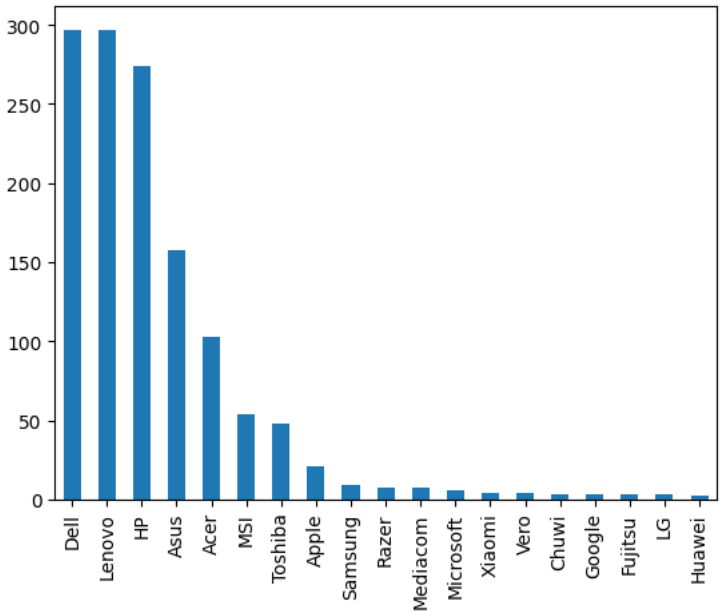
```
<ipython-input-18-87e11caeb2c4>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Price'])
<Axes: xlabel='Price', ylabel='Density'>
```



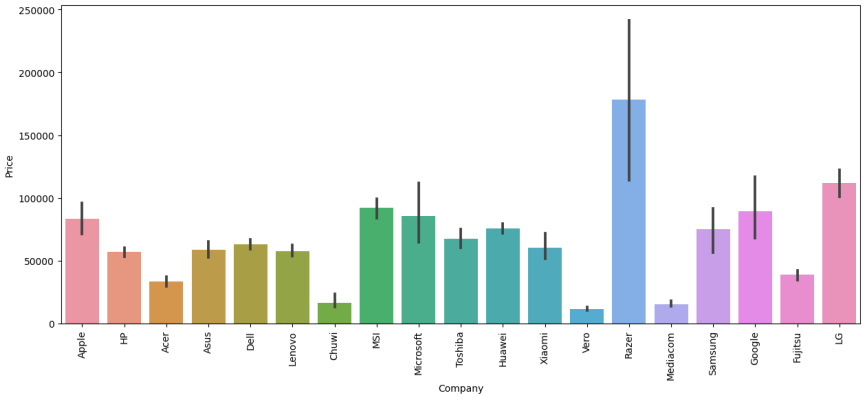
Here we analysed that there are very few laptops having price more than 2Lakh

```
#Plot for the counting of products for each company
df['Company'].value_counts().plot(kind='bar')
```

```
<Axes: >
```

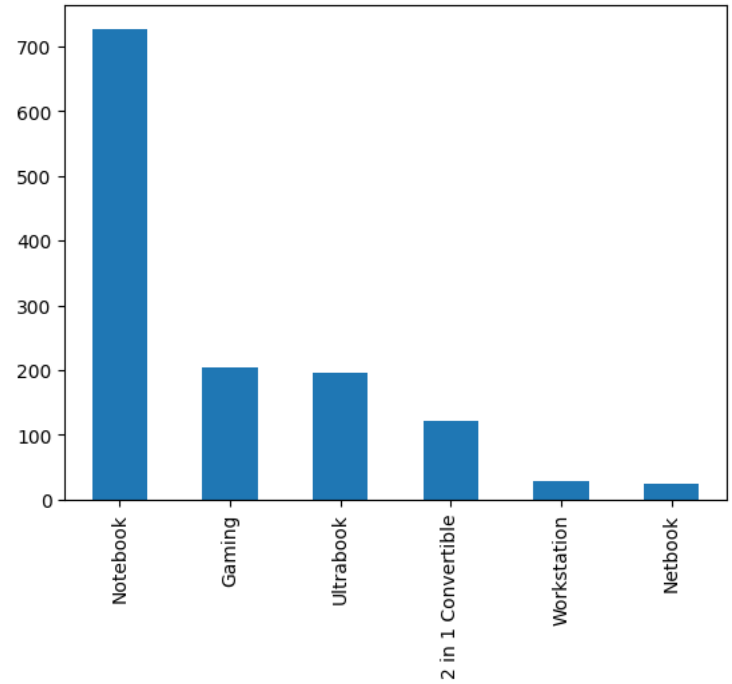



```
#Average price for each of the company
plt.figure(figsize=(15, 6))
sns.barplot(x=df['Company'], y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



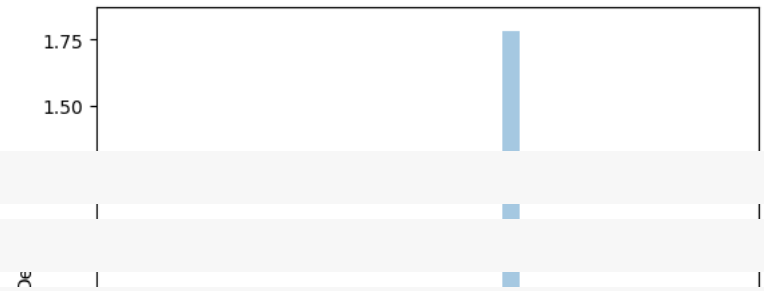

```
df['TypeName'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
sns.distplot(df['Inches'])
```

```
<ipython-input-22-51888cb550e6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Inches'])
<Axes: xlabel='Inches', ylabel='Density'>
```




```
df['ScreenResolution'].value_counts()
```

```
Full HD 1920x1080                                507
1366x768                                         281
IPS Panel Full HD 1920x1080                      230
IPS Panel Full HD / Touchscreen 1920x1080         53
Full HD / Touchscreen 1920x1080                   47
1600x900                                          23
Touchscreen 1366x768                              16
Quad HD+ / Touchscreen 3200x1800                  15
IPS Panel 4K Ultra HD 3840x2160                   12
IPS Panel 4K Ultra HD / Touchscreen 3840x2160     11
4K Ultra HD / Touchscreen 3840x2160               10
4K Ultra HD 3840x2160                              7
Touchscreen 2560x1440                              7
IPS Panel 1366x768                                 7
IPS Panel Quad HD+ / Touchscreen 3200x1800         6
IPS Panel Retina Display 2560x1600                 6
IPS Panel Retina Display 2304x1440                 6
Touchscreen 2256x1504                              6
IPS Panel Touchscreen 2560x1440                    5
IPS Panel Retina Display 2880x1800                 4
IPS Panel Touchscreen 1920x1200                    4
1440x900                                           4
IPS Panel 2560x1440                                4
IPS Panel Quad HD+ 2560x1440                       3
Quad HD+ 3200x1800                                 3
1920x1080                                          3
Touchscreen 2400x1600                              3
2560x1440                                          3
IPS Panel Touchscreen 1366x768                     3
IPS Panel Touchscreen / 4K Ultra HD 3840x2160      2
IPS Panel Full HD 2160x1440                        2
IPS Panel Quad HD+ 3200x1800                       2
IPS Panel Retina Display 2736x1824                 1
IPS Panel Full HD 1920x1200                        1
IPS Panel Full HD 2560x1440                        1
IPS Panel Full HD 1366x768                         1
Touchscreen / Full HD 1920x1080                    1
Touchscreen / Quad HD+ 3200x1800                   1
Touchscreen / 4K Ultra HD 3840x2160                1
IPS Panel Touchscreen 2400x1600                    1
Name: ScreenResolution, dtype: int64
```

```
#Now making new column namely TouchScreen from the column ScreenResolution:
df['TouchScreen'] = df['ScreenResolution'].apply(lambda x:1 if 'Touchscreen' in x else 0)
df.head()
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS

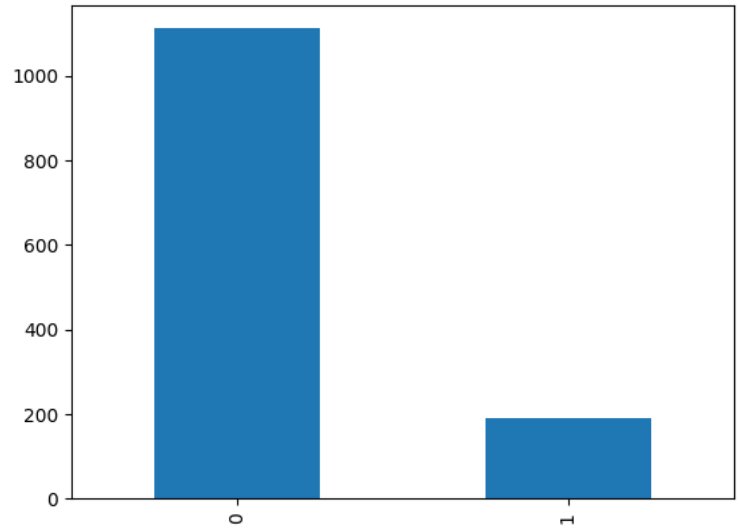
```
df.shape
```

```
(1303, 12)
```

					Intel			Intel HD	
					2.5GHz				

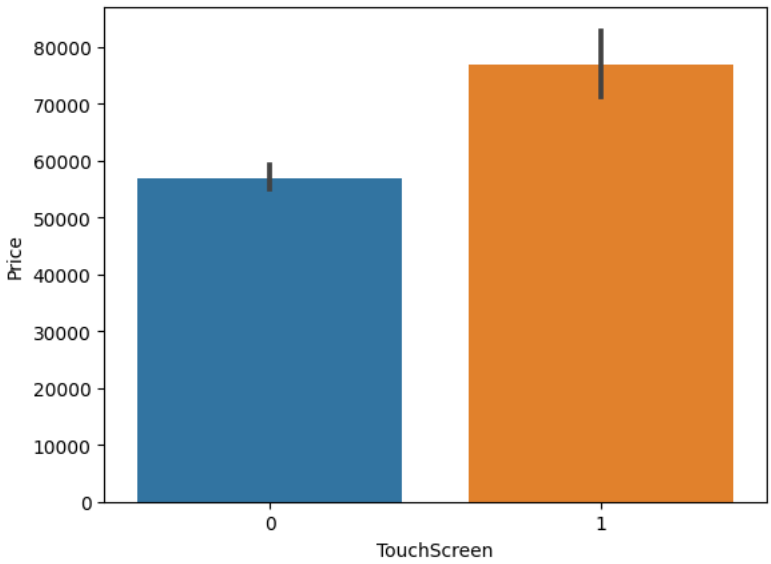
```
df['TouchScreen'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
sns.barplot(x=df['TouchScreen'], y=df['Price'])
```

```
<Axes: xlabel='TouchScreen', ylabel='Price'>
```




```
#Similarly making new column namely IpsPanel from the column ScreenResolution:
df['IpsPanel'] = df['ScreenResolution'].apply(lambda x:1 if 'IPS Panel' in x else 0)
df.head()
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS

```
#Now trying to extract X and Y Resolution:
new = df["ScreenResolution"].str.split('x', n=1, expand=True)
new.head()
```

	0	1
0	IPS Panel Retina Display 2560	1600
1	1440	900
2	Full HD 1920	1080
3	IPS Panel Retina Display 2880	1800
4	IPS Panel Retina Display 2560	1600

```
df['X_res'] = new[0]
df['Y_res'] = new[1]
df.head(2)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5	8	128GB Flash	Intel HD Graphics	macOS

```
#Now applying RegEx on X_res to extract the 3 or more digits:
df['X_res'] = df['X_res'].str.replace(',', '').str.findall(r'(\d+\.?\d+)').apply(lambda x:x[0])
df.head(4)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7	16	512GB SSD	AMD Radeon	macOS

```
#Now changing the Data-types of both X_res and Y_res to int from object
df['X_res'] = df['X_res'].astype('int')
df['Y_res'] = df['Y_res'].astype('int')
df.dtypes
```

```
Company              object
TypeName             object
Inches              float64
ScreenResolution     object
Cpu                  object
Ram                   int64
Memory               object
Gpu                  object
OpSys                object
Weight              float64
Price               float64
TouchScreen           int64
IpsPanel              int64
X_res                 int64
Y_res                 int64
dtype: object
```

```
#Checking the Correlation
df.corr(numeric_only=True)
```

	Inches	Ram	Weight	Price	TouchScreen	IpsPanel	X_res
Inches	1.000000	0.237993	0.827631	0.068197	-0.361735	-0.114804	-0.071245
Ram	0.237993	1.000000	0.383874	0.743007	0.116984	0.206623	0.433121
Weight	0.827631	0.383874	1.000000	0.210370	-0.294620	0.016967	-0.032880
Price	0.068197	0.743007	0.210370	1.000000	0.191226	0.252208	0.556529
TouchScreen	-0.361735	0.116984	-0.294620	0.191226	1.000000	0.150512	0.351066
IpsPanel	-0.114804	0.206623	0.016967	0.252208	0.150512	1.000000	0.281457
X_res	-0.071245	0.433121	-0.032880	0.556529	0.351066	0.281457	1.000000
Y_res	-0.095404	0.424437	-0.052946	0.552809	0.357076	0.280930	0.994219

```
df.corr(numeric_only=True)['Price']
```

```
Inches         0.068197
Ram            0.743007
Weight         0.210370
Price          1.000000
TouchScreen    0.191226
IpsPanel       0.252208
X_res          0.556529
Y_res          0.552809
Name: Price, dtype: float64
```

```
#Now by using Inches, X_res and Y_res , make new column PPI (pixels per inch)--
df['ppi'] = (((df['X_res']**2 + df['Y_res']**2))**0.5/df['Inches']).astype('float')
```

```
df.head(4)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys
				IPS Panel Retina	Intel		128GB	Intel Iris Plus	

```
#Now checking the correlation of ppi with price:
df.corr(numeric_only=True)['Price']
```

```
Inches         0.068197
Ram            0.743007
Weight         0.210370
Price          1.000000
TouchScreen    0.191226
IpsPanel       0.252208
X_res          0.556529
Y_res          0.552809
ppi            0.473487
Name: Price, dtype: float64
```

Hence we observed that ppi has good relation with price;

```
#Now we don't Require 'ScreenResolution' ,  'X_res' , 'Y_res' and 'Inches' column bcz we extracted all the features (ppi) from it;
df.drop(columns=['ScreenResolution', 'X_res', 'Y_res', 'Inches'], inplace=True)
```

```
df.head(2)
```

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Touch
0	Apple	Ultrabook	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	

```
df['Cpu'].value_counts()
```

```
Intel Core i5 7200U 2.5GHz       190
Intel Core i7 7700HQ 2.8GHz      146
Intel Core i7 7500U 2.7GHz       134
Intel Core i7 8550U 1.8GHz        73
Intel Core i5 8250U 1.6GHz        72
                                ...
Intel Core M M3-6Y30 0.9GHz        1
AMD A9-Series 9420 2.9GHz          1
Intel Core i3 6006U 2.2GHz         1
AMD A6-Series 7310 2GHz            1
Intel Xeon E3-1535M v6 3.1GHz      1
Name: Cpu, Length: 118, dtype: int64
```

Now extracting the important features from Cpu column

```
df['Cpu_name'] = df['Cpu'].apply(lambda x:" ".join(x.split()[0:3]))
df.sample(5)
```

	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price
328	HP	Ultrabook	Intel Core i7	8	256GB	Intel HD Graphics	Windows	1.26	71128.8000

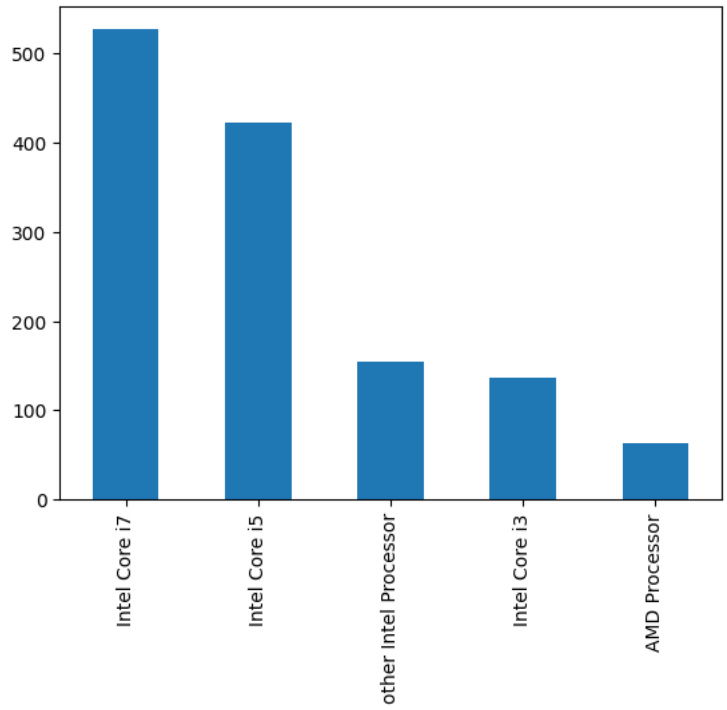
```
def fetch_processor(text):
    if text == 'Intel Core i3' or  text == 'Intel Core i5' or  text == 'Intel Core i7':
        return text

    else:
        if text.split()[0] == 'Intel':
            return "other Intel Processor"

        else:
            return "AMD Processor"
```

```
df['Cpu Brand'] = df['Cpu_name'].apply(fetch_processor)
df.sample(4)
```

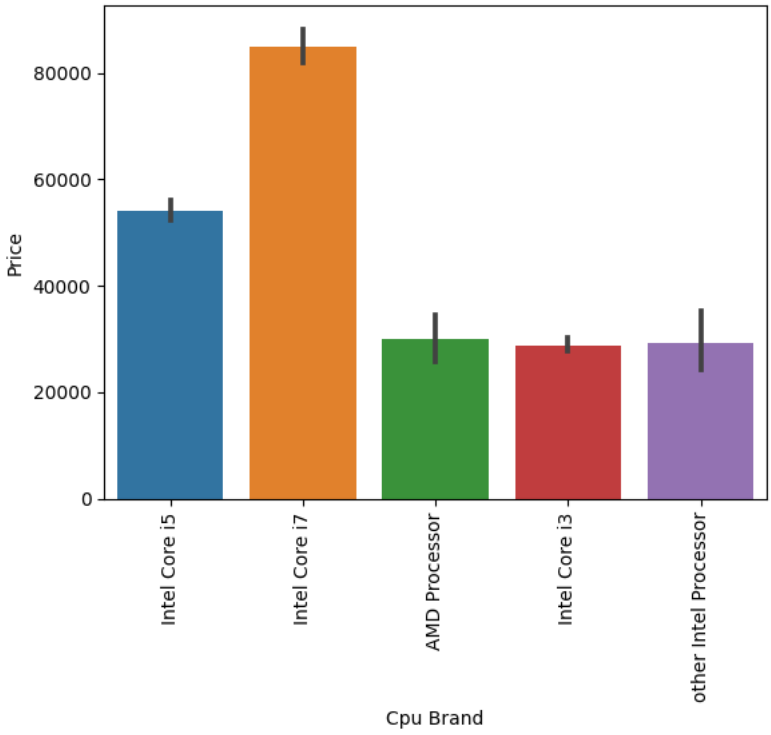
	Company	TypeName	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price	Tou
200	Dell	Gaming	Intel Core i7 7700HQ 2.8GHz	16	512GB SSD + 1TB HDD	Nvidia GeForce GTX 1060	Windows 10	2.65	98301.60	
134	HP	Notebook	Intel Core i7 7500U 2.7GHz	8	1TB HDD	Intel HD Graphics 620	Windows 10	2.05	31861.44	
568	Lenovo	Notebook	Intel Pentium Quad Core	4	500GB HDD	Intel HD Graphics 505	Windows 10	2.20	18328.32	

```
df['Cpu Brand'].value_counts().plot(kind='bar')
```

```
<Axes: >
```




```
#Variation of price with brand:
sns.barplot(x=df['Cpu Brand'], y=df['Price'])
plt.xticks(rotation='vertical')
plt.show()
```



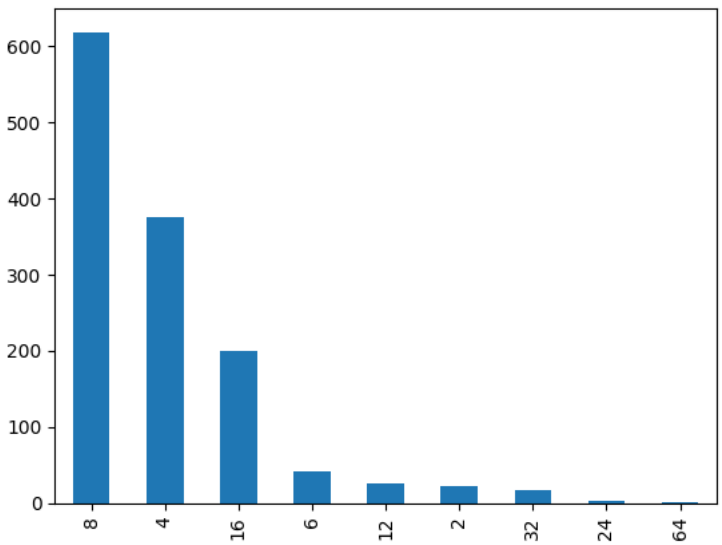

```
#Now there is no need of columns Cpu and Cpu_names, so drop them:
df.drop(columns=['Cpu', 'Cpu_name'], inplace=True)
```

```
df.head()
```

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchScreen
0	Apple	Ultrabook	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0
1	Apple	Ultrabook	8	128GB Flash	Intel HD Graphics	macOS	1.34	47895.5232	0

```
df['Ram'].value_counts().plot(kind='bar')
```

```
<Axes: >
```



```
df['Memory'].value_counts()
```

```
256GB SSD                        412
1TB HDD                          223
500GB HDD                        132
512GB SSD                        118
128GB SSD +  1TB HDD              94
128GB SSD                         76
256GB SSD +  1TB HDD              73
32GB Flash Storage                38
2TB HDD                           16
64GB Flash Storage                15
512GB SSD +  1TB HDD              14
1TB SSD                           14
256GB SSD +  2TB HDD              10
1.0TB Hybrid                       9
256GB Flash Storage                8
16GB Flash Storage                 7
32GB SSD                           6
180GB SSD                          5
128GB Flash Storage                4
512GB SSD +  2TB HDD               3
16GB SSD                           3
512GB Flash Storage                2
1TB SSD +  1TB HDD                 2
256GB SSD +  500GB HDD             2
128GB SSD +  2TB HDD               2
256GB SSD +  256GB SSD             2
512GB SSD +  256GB SSD             1
512GB SSD +  512GB SSD             1
64GB Flash Storage +  1TB HDD      1
1TB HDD +  1TB HDD                 1
32GB HDD                           1
64GB SSD                           1
128GB HDD                          1
240GB SSD                          1
8GB SSD                            1
508GB Hybrid                       1
1.0TB HDD                          1
512GB SSD +  1.0TB Hybrid          1
256GB SSD +  1.0TB Hybrid          1
Name: Memory, dtype: int64
```

```
df['Memory'] = df['Memory'].astype(str).replace('\.0', '', regex=True)
df["Memory"] = df["Memory"].str.replace('GB', '')
df["Memory"] = df["Memory"].str.replace('TB', '000')
new = df["Memory"].str.split("+", n = 1, expand = True)

df["first"]= new[0]
df["first"]=df["first"].str.strip()

df["second"]= new[1]

df["Layer1HDD"] = df["first"].apply(lambda x: 1 if "HDD" in x else 0)
df["Layer1SSD"] = df["first"].apply(lambda x: 1 if "SSD" in x else 0)
df["Layer1Hybrid"] = df["first"].apply(lambda x: 1 if "Hybrid" in x else 0)
df["Layer1Flash_Storage"] = df["first"].apply(lambda x: 1 if "Flash Storage" in x else 0)

df['first'] = df['first'].str.replace(r'\D', '', regex=True)

df["second"].fillna("0", inplace = True)

df["Layer2HDD"] = df["second"].apply(lambda x: 1 if "HDD" in x else 0)
df["Layer2SSD"] = df["second"].apply(lambda x: 1 if "SSD" in x else 0)
df["Layer2Hybrid"] = df["second"].apply(lambda x: 1 if "Hybrid" in x else 0)
df["Layer2Flash_Storage"] = df["second"].apply(lambda x: 1 if "Flash Storage" in x else 0)

df['second'] = df['second'].str.replace(r'\D', '', regex=True)

df["first"] = df["first"].astype(int)
df["second"] = df["second"].astype(int)

df["HDD"]=(df["first"]*df["Layer1HDD"]+df["second"]*df["Layer2HDD"])
df["SSD"]=(df["first"]*df["Layer1SSD"]+df["second"]*df["Layer2SSD"])
df["Hybrid"]=(df["first"]*df["Layer1Hybrid"]+df["second"]*df["Layer2Hybrid"])
df["Flash_Storage"]=(df["first"]*df["Layer1Flash_Storage"]+df["second"]*df["Layer2Flash_Storage"])

df.drop(columns=['first', 'second', 'Layer1HDD', 'Layer1SSD', 'Layer1Hybrid',
       'Layer1Flash_Storage', 'Layer2HDD', 'Layer2SSD', 'Layer2Hybrid',
       'Layer2Flash_Storage'],inplace=True)
```

```
df.sample(5)
```

	Company	TypeName	Ram	Memory	Gpu	OpSys	Weight	Price	TouchS
429	Mediacom	2 in 1 Convertible	4	32 SSD	Intel HD Graphics 500	Windows 10	1.16	15930.7200	
1106	MSI	Gaming	8	128 SSD + 1000 HDD	Nvidia GeForce GTX 960M	Windows 10	2.90	80516.2032	
1286	Lenovo	Notebook	2	64 Flash Storage	Intel HD Graphics	Windows 10	1.50	12201.1200	
893	Lenovo	Ultrabook	8	256 SSD	Intel HD Graphics 620	Windows 10	1.32	95850.7200	
1179	HP	Notebook	4	500 HDD	Intel HD Graphics 520	Windows 10	2.07	34632.0000	

```
df.drop(columns=['Memory'], inplace=True)
df.head(2)
```

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	TouchScreen	IpsPanel
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1

```
df.corr(numeric_only=True)['Price']
```

```
Ram              0.743007
Weight           0.210370
Price            1.000000
TouchScreen      0.191226
IpsPanel         0.252208
ppi              0.473487
HDD             -0.096441
SSD              0.670799
Hybrid           0.007989
Flash_Storage   -0.040511
Name: Price, dtype: float64
```

```
df.drop(columns=['Hybrid', 'Flash_Storage'], inplace=True)
df.head(3)
```

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	TouchScreen	IpsPanel
0	Apple	Ultrabook	8	Intel Iris Plus Graphics	macOS	1.37	71378.6832	0	1

```
#Now analysing the Gpu columns:
df['Gpu'].value_counts()
```

```
Intel HD Graphics 620      281
Intel HD Graphics 520      185
Intel UHD Graphics 620      68
Nvidia GeForce GTX 1050     66
Nvidia GeForce GTX 1060     48
                          ...
AMD Radeon R5 520            1
AMD Radeon R7                1
Intel HD Graphics 540        1
AMD Radeon 540               1
ARM Mali T860 MP4            1
Name: Gpu, Length: 110, dtype: int64
```

```
df['Gpu brand'] = df['Gpu'].apply(lambda x:x.split()[0])
df.head(4)
```

	Company	TypeName	Ram	Gpu	OpSys	Weight	Price	TouchScreen	IpsPanel
0	Apple	Ultrabook	8	Intel Iris Plus Graphics 640	macOS	1.37	71378.6832	0	1

```
df['Gpu brand'].value_counts()
```

```
Intel     722
Nvidia    400
AMD       180
ARM         1
Name: Gpu brand, dtype: int64
```

```
df = df[df['Gpu brand'] != 'ARM']
df['Gpu brand'].value_counts()
```

```
Intel     722
Nvidia    400
AMD       180
Name: Gpu brand, dtype: int64
```

```
df.drop(columns=['Gpu'], inplace=True)
```

```
<ipython-input-59-02e14c40d970>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df.drop(columns=['Gpu'], inplace=True)
```

```
df.sample()
```

Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IpsPanel

```
df['OpSys'].value_counts()
```

```
Windows 10      1072
No OS             66
Linux             62
Windows 7         45
Chrome OS         26
macOS             13
Mac OS X           8
Windows 10 S       8
Android            2
Name: OpSys, dtype: int64
```

```
def cat_os(inp):
    if inp == 'Windows 10' or  inp == 'Windows 7' or  inp == 'Windows 10 S':
        return 'Windows'

    elif inp == "macOS" or inp == "Mac OS X":
        return "Mac"
    else:
        return 'Others'
```

```
df['os'] = df['OpSys'].apply(cat_os)
```

```
<ipython-input-63-38671a3c07bd>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['os'] = df['OpSys'].apply(cat_os)
```

```
df.head()
```

	Company	TypeName	Ram	OpSys	Weight	Price	TouchScreen	IpsPanel	
0	Apple	Ultrabook	8	macOS	1.37	71378.6832	0	1	226.983
1	Apple	Ultrabook	8	macOS	1.34	47895.5232	0	0	127.677

```
df.drop(columns=['OpSys'], inplace=True)
```

```
df.head(3)
```

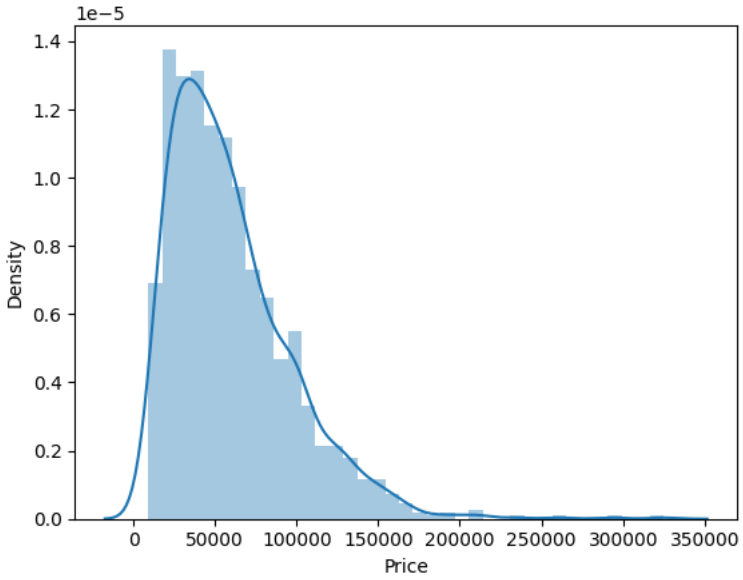
	Company	TypeName	Ram	Weight	Price	TouchScreen	IpsPanel	ppi	Cp Bran
0	Apple	Ultrabook	8	1.37	71378.6832	0	1	226.983005	Inte Cor

```
df.corr(numeric_only=True)
```

	Ram	Weight	Price	TouchScreen	IpsPanel	ppi	HDD
Ram	1.000000	0.383362	0.742905	0.118875	0.207949	0.305688	0.095808
Weight	0.383362	1.000000	0.209867	-0.293004	0.018643	-0.321883	0.514147
Price	0.742905	0.209867	1.000000	0.192917	0.253320	0.475368	-0.096891
TouchScreen	0.118875	-0.293004	0.192917	1.000000	0.148026	0.458571	-0.208766
IpsPanel	0.207949	0.018643	0.253320	0.148026	1.000000	0.299142	-0.093588
ppi	0.305688	-0.321883	0.475368	0.458571	0.299142	1.000000	-0.294698
HDD	0.095808	0.514147	-0.096891	-0.208766	-0.093588	-0.294698	1.000000

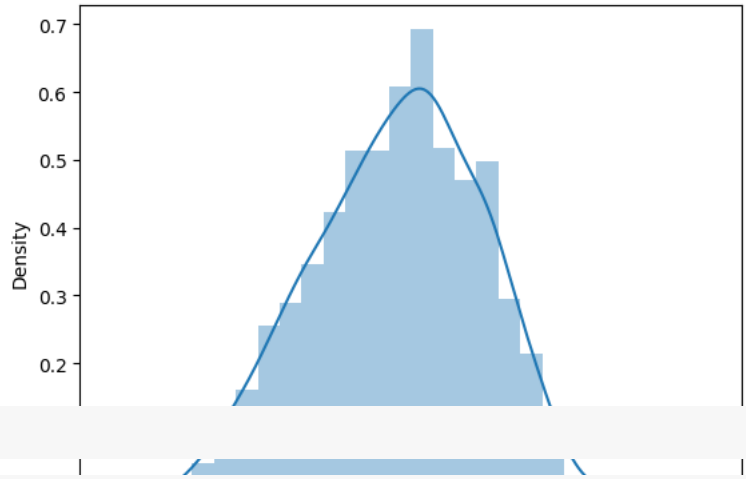
```
sns.distplot(df['Price'])
```

```
<ipython-input-68-87e11caeb2c4>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Price'])
<Axes: xlabel='Price', ylabel='Density'>
```




```
sns.distplot(np.log(df['Price']))
```

```
<ipython-input-69-c1a82a4801f0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(np.log(df['Price']))
<Axes: xlabel='Price', ylabel='Density'>
```




```
X = df.drop(columns=['Price'])
y = np.log(df['Price'])
X
```

	Company	TypeName	Ram	Weight	TouchScreen	IpsPanel	ppi	Cpu Brand
0	Apple	Ultrabook	8	1.37	0	1	226.983005	Intel Core i5
1	Apple	Ultrabook	8	1.34	0	0	127.677940	Intel Core i5
2	HP	Notebook	8	1.86	0	0	141.211998	Intel Core i5
3	Apple	Ultrabook	16	1.83	0	1	220.534624	Intel Core i7
4	Apple	Ultrabook	8	1.37	0	1	226.983005	Intel Core i5
...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	4	1.80	1	1	157.350512	Intel Core i7

```
y
```

```
0       11.175755
1       10.776777
2       10.329931
3       11.814476
4       11.473101
          ...
1298    10.433899
1299    11.288115
1300     9.409283
1301    10.614129
1302     9.886358
Name: Price, Length: 1302, dtype: float64
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=2)
```

```
#Handling Object type cols:
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score, mean_absolute_error
```

```
from sklearn.ensemble import RandomForestRegressor
```

Random Forest

```
step1 = ColumnTransformer(transformers=[
    ('col_tnf', OneHotEncoder(sparse=False, drop='first'), [0,1,7,10,11])
], remainder='passthrough')

step2 = RandomForestRegressor(n_estimators=100,
                              random_state=3,
                              max_samples=0.5,
                              max_features=0.75,
                              max_depth=15)

pipe = Pipeline([
    ('step1', step1),
    ('step2', step2)
])

pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)

print('R2 Score: ', r2_score(y_test, y_pred))
print('Mean Absolute Error: ', mean_absolute_error(y_test, y_pred))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_outpu
  warnings.warn(
R2 Score:  0.8873402378382488
Mean Absolute Error:  0.15860130110457718
```